Finger Tracking, Take 2

CIS (4|5)22, Fall 2009

Michal Young

Version 1.1 of 2009.10.01

Abstract

The project for Fall 2009 (for the whole term, or optionally for the first half) is a system for recording, analyzing, and displaying traces of finger movement on a tactile map. We can start from proof-of-concept prototypes constructed by a class in Spring 2009. The most important objective is to design a good software architecture and build a solid foundation for further development, but there are several possibilities for improving functionality and adding or enhancing features.

This document briefly describes what a finger-tracking system is, why it is being constructed, what has already been constructed and demonstrated, and what I hope to accomplish in Fall 2009.

1 Introduction

1.1 Why?

You may have the illusion of being able to see a large part of the room, or a page, all at once. We know, however, that this is an illusion: Actually your eye is able to see details within only 2-3 degrees of visual angle (roughly equivalent to your thumbnail, held at arm's length); beyond that you can pick up a little bit of information in the parafoveal region extending to about 6 degrees of visual angle, and in your peripheral vision you can detect motion but little else. When we read, the eye moves in jerks, forward and sometimes back, allowing us to pick up a few characters at a time, despite our impression of scanning smoothly from left to right.

How do we know about the fovea and the way reading is performed through a series of visual snapshots? A good deal of our understanding of visual perception, and especially of reading, was gained through experiments with eye tracking equipment. For example, researchers in reading have devised clever experiments in which the actual text of a passage is displayed within the foveal region, and random characters are substituted elsewhere.

What we know about visual perception is important for the design of both interactive and non-interactive media, like computer user interfaces and printed maps. But what if we are designing media (like maps) for people who are blind or visually impaired, and must use other senses in place of vision?

The purpose of a *finger tracker* is to provide researchers the ability to record and analyze the finger motions of blind users who read with their fingers. I know of only a couple previous finger tracking research instruments, and those have been developed primarily for research on how blind people read Braille text. I do not know of any previous research instruments for analyzing the finger motions used by blind people reading tactile maps. Do they use one hand to maintain a "base" location and explore around it with the other? Do they use different scanning strategies depending on whether they are planning a route or gathering other information? Map researchers have some ideas based on observation, but they have not so far had access to instruments that can record finger movement data for analysis.

It is easy to imagine other uses of finger tracking as well. Several researchers have experimented with finger tracking as a form of input in graphical user interfaces, such as the "digital desk" [1] or the "SixthSense" augmented reality project [2].

1.2 What?

An initial "proof-of-concept" prototype of a finger-tracking system was constructed by students in CIS 423/523, Spring 2009. Figure 1 sketches, very roughly, the components and overall flow of that prototype: An input component (video capture and extract, or tablet capture) produces a sequence of records; both input components produce an XML file in the same format. Each record is one point in the recording, and includes x and y coordinates, a time-stamp, and an indication of which finger is being recorded. This point-stream file is interpreted by an analysis and display component.

In the initial prototype, the tablet capture module uses a standard Wacom Intuos graphics tablet. Students disassembled an Intuos stylus and fastened it to a thin glove with velcro. (It is not necessary for the stylus to actually touch the tablet, so the pen electronics can be mounted on the back side of a finger.) Since the tablet available to students is capable of capturing one point at a time (i.e., it is not a multi-touch device), the tablet module produces a sequence of points for a single finger. Pressure and tilt sensors are not used. The tablet capture component, which is packaged as a standalone Java program, produces accurate and clean (not noisy) sequence of time-stamped position records at a rapid rate.

Video recording (with a standard video camera) and extraction from the video are distinct steps in the initial prototype. Video recording produces video in a file,

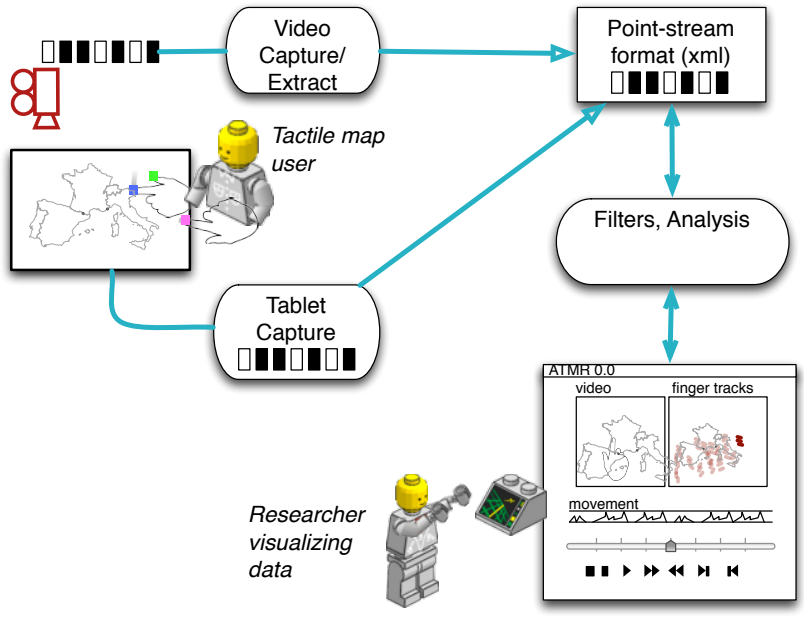

Figure 1: Main components of finger tracking

in a standard format. To produce usable video, brightly colored stickers are attached to one or more fingers. Video extraction is applied to that file. Initially the colors to be tracked are identified, and then the extraction component uses image processing techniques to track the position of the colored patches in each frame of the video to produce a sequence of time-stamped position records. A pleasant surprise of the spring project was the speed of image processing. An unpleasant surprise was the multitude of incompatible MPEG video formats. Although the image processing algorithms ran quickly enough that they could have been applied in real-time while shooting video, in practice a separate program was used to convert files from a format we could produce with the camera to a different format that was acceptable to the image processing libraries we used.

The third component in the original system provided visualization and management of data from the input components. Logically this is divisible into distinct parts: Analysis and transformation of the traces (e.g., annotating a trace with velocities of finger motion, calculated from sequences of position records), visualization and display (including the original video), and management of the experimental data. In the first prototype, these features were combined into one program.

2 Goals

The Spring 2009 project was very much a "proof of concept." We needed to know whether a finger tracking system using a graphics tablet and/or video was practical, and roughly how difficult it would be. Also, since there are few finger-tracking systems for research use, and no prior finger trackers (as far as we know) for studying how blind people read tactile maps, a prototype was a useful step in understanding researchers' requirements for such a system. Even a very basic prototype can be useful for exploring ideas about, for example, possible visualizations of finger-tracking data, and is likely to reveal important issues that might never become apparent otherwise.

Every software project involves trade-offs. A prototype that is intended only to help users and developers understand requirements may compromise every other quality to maximize speed of delivery. Such a protytpe is a *throwaway*. Building something very, very quickly was a high priority for the spring 2009 project, but it is not a throwaway. Despite many compromises and short-cuts, it was intended to be a starting point for new projects. By providing some implementation of each of the basic steps in finger tracking — capture, analysis, and visualization — it permits the next set of developers to focus on enhancing or redeveloping one part while using other parts as they are, however imperfect.

For Fall 2009, I have two goals: to improve the overall *architecture* of the finger tracking system, and to make functional enhancements.

Architecture or *architectural design* refers to the overall structure of the system — how it is divided into pieces, how the pieces fit together, the overall pattern that tells us how pieces can be changed, added, removed, and replaced. Architectural design is very near the core of software engineering as a discipline, and is deeply intertwined with other issues in software development, from requirements engineering to software process to validation and verification. Very roughly speaking, the *architectural* aspect of architectural design comes from viewing the system as one of a family or series of systems, designing for the whole family in the context of its development. We will talk a *lot* about architectural design throughout the term.

Several functional enhancements to the finger tracking system are possible. My wish list includes:

- Real-time finger tracking with video. For the intended use in research, recording video and then later extracting finger tracks is acceptable. However, there are several potential advantages in tracking finger motion in real-time, while the video is being recorded. The greatest of these is reducing the likelihood of wasting an experimental session, producing unusable video. Experimen-

tal time with subjects is precious, and one cannot just repeat a session if one is spoiled (repeating a session with a subject is not the same experimental condition as the initial experiment). The image processing steps in finger tracking are very sensitive to lighting, and may also be subject to other problems. If the image analysis is being performed during the video shoot, then the researcher can be alerted immediately if conditions are poor, and make adjustments.

A second motivation for finger tracking in real-time is that, in addition to a research instrument, video finger tracking may be useful as an input mode for interactive maps for blind persons. Previous CIS 422/522 classes have developed *soundscape* maps for blind users, substituting sound for graphics. A major challenge in soundscape maps is maintaining orientation, i.e., a sense of where the currently explored position is in the overall map (see Figure 2). One possible approach to orientation is to exploit the proprioceptive sense of limb position, using video to place the virtual map surface on a large surface like a table-top.

- Visualization interfaces. The Spring 2009 prototype includes some sample visualizations of finger traces, as an example for designing and building more. In addition to those, it would be very nice to have interfaces to other visualization and analysis tools, in particular *R* statistical processing language [4] and/or the *Processing* visualization environment [3]. It is even possible that one of these should replace the current visualization facility, although I would prefer to maintain multiple options for creating visualizations.

- Image registration. We have been told that users of tactile maps often turn and move them while reading. Image registration means reorienting the image accordingly. While image registration is a well-studied problem, and can be made easier by using distinctive graphical marks on the map, it raises other interesting design issues. Consider, for example, analysis of changes in the velocity of a finger (analogous to study of saccades in eye tracking). Should we consider motion relative to the camera (which is fixed), or relative to the map (which may be moving)? One can imagine uses for both sets of data, so image registration probably requires some rethinking of the point stream data format to identify whether the (x, y) coordinates are registered or unregistered.

This list of potential enhancements is not exhaustive, and I hope for some creative and interesting ideas. Our primary user (Professor Amy Lobben in Geography) will have a very busy term, but if possible we will try to get feedback from

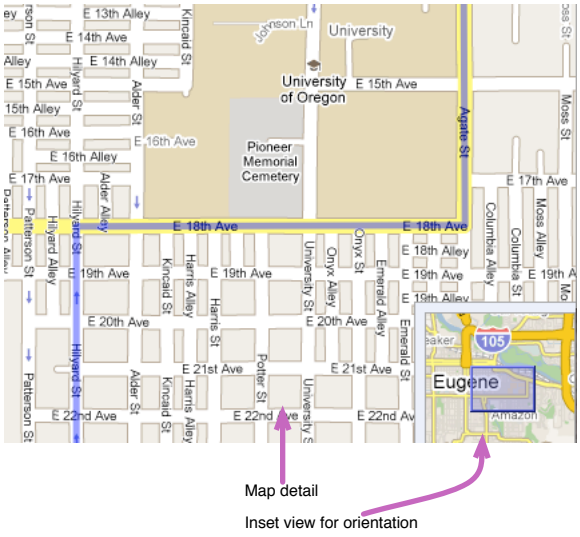

Figure 2: Google maps provide an inset (lower right corner) to help the (sighted) map user *orient* the current zoomed-in map view with respect to a larger map area. Orientation is an important issue in designing interactive maps for blind users.

her at least once during the term, and certainly we will invite her to project presentations at the end of the term.

3 Schedule and Constraints

There are two project deadlines in CIS 422/522. You have the option of carrying the initial project through to the end of the term, which is what I recommend. However, I know that some students come to CIS 422/522 with a strong desire to complete a different project, perhaps with a team of their own choosing. I will attempt to accommodate that. I will need to know right away if that is your plan, so that I can form initial teams appropriately.

The first project turn-in is about half-way through the term; I will make it 5pm on October 23, the end of week 4. The second (final) project deadline is *before* dead week — which in Fall term means it must be before the Thanksgiving holiday weekend. In fact, I will make it due on Tuesday, November 24 at 5pm to avoid constraining holiday travel plans. During dead week (November 30 and December 2) you will make public project presentations during class time.

We really have only about 8 weeks total time to work on projects. This makes it very hard to complete two *different* projects. I have seen teams try to do so and

succeed wonderfully. I have seen teams try to do so and fail catastrophically. Try it at your own risk.

The content of the first turn-in should be a working system, whether or not you are continuing the project through to the end of the term. If you will pursue a different project in the second half of the term, then the first turn-in is the final turn-in for the project — you can (and should) scale down your ambitions appropriately to the time available, but you cannot turn in a half-baked mess of non-working code and missing documentation. If you are continuing the project, it may be appropriate to postpone completion of some artifacts in order to focus on riskier, more critical parts of the project, but you should still turn in a working (even if incomplete) system, and you should make good software engineering arguments for your decisions of what to accelerate and what to postpone.

4 How to start

4.1 Team formation

I will hand out a questionnaire and collect it on the first day of class. I will do my best to send you your team assignment by the next day. We do not have class on Friday, but that may be a good day to arrange an initial team meeting. It is important to quickly work out some basics like possible meeting times and preferred modes of communication (both synchronous and asynchronous). If there are problems in my assignments of team members, I need to know *right away* to have any chance to make adjustments.

4.2 Studying the code

An important first task — and not an easy one — is studying the code and documentation produced by students last spring. It is important to get an overall view of what the major pieces were and how they fit together, and then some of the technical details like how the image processing libraries were used. It is hard to make rational decisions about what you will change until you understand a bit about what already exists. You will probably need to go back and forth between looking at actual code and looking at notes and documentation from the developers. The developers hosted the project on the Assembla service at `http://www.assembla.com/wiki/show/atmr`.

4.3 Establish your infrastructure and learn to use it

You will need (at least) a version control system and communication. You could set up everything on the CS systems at UO, but in recent terms student teams have found it most convenient to use a hosting service. I encourage you to use Assembla, because it may be easier to coordinate if everyone is using the same service. Assembla provides subversion for files, a wiki for communication and collaborative document building, and a simple "ticket" system for project management.

4.4 Start planning

The first thing I will ask you to do, in a class presentation Wednesday, is describe tentatively how you plan to tackle the project. How will you divide up the work? What do you hope to accomplish by the first turnin? How will you refine your plan? I expect this to be pretty rough ... but try to get beyond "I have no idea", and see what you can do. The presentation should be short (about 10 minutes), leaving us time to talk both about this project, about the difficulties you encounter, and how one makes a plan with risks and imperfect information.

References

[1] BROWN, T., AND THOMAS, R. C. Finger tracking for the digital desk. *Australasian User Interface Conference 0* (2000), 11.

[2] MISTRY, P., MAES, P., AND CHANG, L. WUW — wear Ur world: a wearable gestural interface. In *CHI EA '09: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2009), ACM, pp. 4111–4116.

[3] Processing (web page). `http://www.processing.org`. Viewed 2009.9.29.

[4] The R project for statistical computing. `http://www.r-project.org/`. Viewed 2009.9.29.

Recent changes

1.1 First presentation Wednesday, not Monday.

1.0 First semi-complete version.